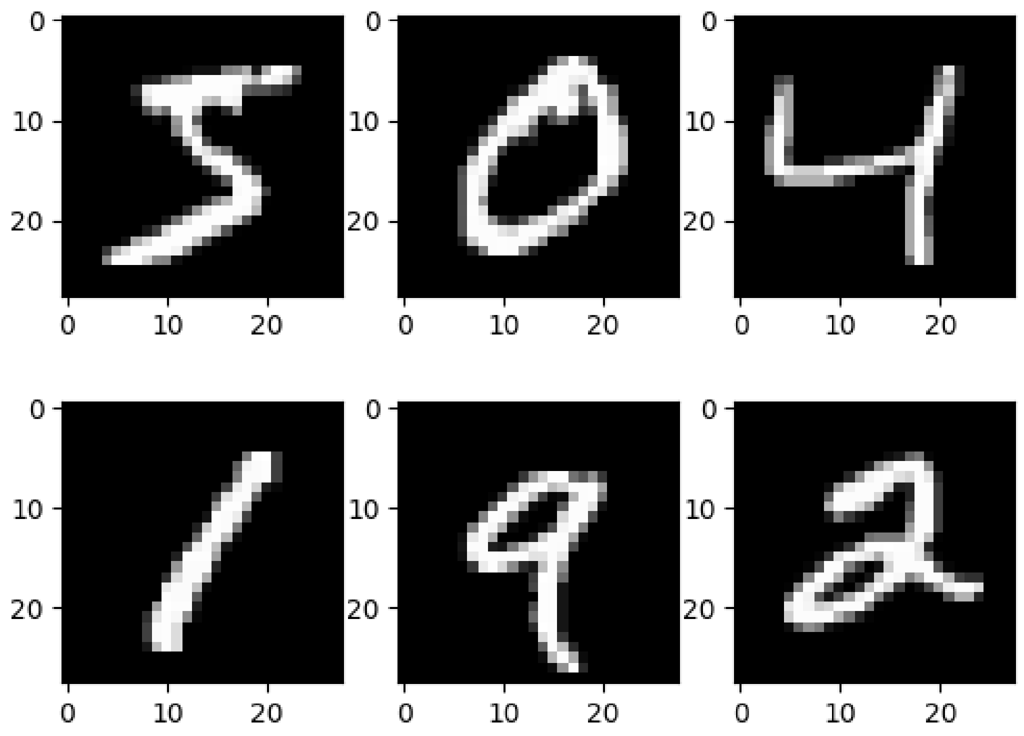
app

October 30, 2024

[10]:
```
#import libraries
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from keras.utils import to_categorical
from keras import backend as k
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from keras.layers import Input
```

[11]:
```
# Load dataset directly from keras library
(X_train, y_train), (X_test, y_test) = mnist.load_data()
```

[12]:
```
# Plot first six samples of MNIST training dataset as grayscale images
for i in range(6):
    plt.subplot(230 + i + 1)
    plt.imshow(X_train[i], cmap=plt.get_cmap('gray'))
plt.show()
```



```
[13]: # Reshape format [sample][width][height][channels]
      X_train = X_train.reshape(X_train.shape[0], 28, 28, 1).astype('float32')
      X_test = X_test.reshape(X_test.shape[0], 28, 28, 1).astype('float32')

      # Convert class vectors (integers) to binary class matrix
      y_train = to_categorical(y_train)
      y_test = to_categorical(y_test)
```

```
[14]: # Normalize inputs
      X_train = X_train / 255
      X_test = X_test / 255
```

```
[15]: # Define input shape
      input_shape = (28, 28, 1)
```

```
[16]: # Define a CNN model
      def create_model():
          num_classes = 10
          model = Sequential()
          model.add(Input(shape=input_shape))  # Specify input shape here
          model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
          model.add(Conv2D(64, (3, 3), activation='relu'))
          model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
    model.add(Dropout(0.25))
    model.add(Flatten())
    model.add(Dense(256, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam',␣
 ↪metrics=['accuracy'])
    return model

# Build the model
model = create_model()
```

[17]:
```
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10,␣
 ↪batch_size=200, verbose=2)
print("The model has successfully trained.")

# Save the model
model.save('model.h5')
print("The model has successfully saved.")
```

```
Epoch 1/10
300/300 - 67s - 223ms/step - accuracy: 0.9311 - loss: 0.2246 - val_accuracy:
0.9836 - val_loss: 0.0524
Epoch 2/10
300/300 - 56s - 188ms/step - accuracy: 0.9796 - loss: 0.0684 - val_accuracy:
0.9895 - val_loss: 0.0342
Epoch 3/10
300/300 - 54s - 181ms/step - accuracy: 0.9842 - loss: 0.0514 - val_accuracy:
0.9902 - val_loss: 0.0288
Epoch 4/10
300/300 - 85s - 282ms/step - accuracy: 0.9875 - loss: 0.0404 - val_accuracy:
0.9907 - val_loss: 0.0291
Epoch 5/10
300/300 - 56s - 188ms/step - accuracy: 0.9894 - loss: 0.0329 - val_accuracy:
0.9897 - val_loss: 0.0310
Epoch 6/10
300/300 - 58s - 192ms/step - accuracy: 0.9906 - loss: 0.0291 - val_accuracy:
0.9917 - val_loss: 0.0258
Epoch 7/10
300/300 - 59s - 196ms/step - accuracy: 0.9920 - loss: 0.0243 - val_accuracy:
0.9917 - val_loss: 0.0272
Epoch 8/10
300/300 - 57s - 192ms/step - accuracy: 0.9928 - loss: 0.0223 - val_accuracy:
0.9909 - val_loss: 0.0279
Epoch 9/10
300/300 - 57s - 190ms/step - accuracy: 0.9938 - loss: 0.0197 - val_accuracy:
```

```
0.9918 - val_loss: 0.0260
Epoch 10/10
300/300 - 79s - 262ms/step - accuracy: 0.9937 - loss: 0.0179 - val_accuracy:
0.9909 - val_loss: 0.0292
```

```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or
`keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g.
`model.save('my_model.keras')` or `keras.saving.save_model(model,
'my_model.keras')`.
```

```
The model has successfully trained.
The model has successfully saved.
```

[21]:
```
# Evaluate the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("CNN error: %.2f%%" % (100 - scores[1] * 100))
```

```
CNN error: 0.91%
```

[ ]:
```
```